Stemming Techniques for Arabic Words: A Comparative Study

May Y. Al-Nashashibi, D. Neagu
Department of Computing, University of Bradford,
Bradford, UK,
e-mail: (M.Y.A.Al-nashashibi,D.Neagu)@bradford.ac.uk

Ali A. Yaghi
Department of Computer Science, Petra University,
Amman, Jordan,
e-mail: Ali.Yaghi@uop.edu.jo

***Abstract*—Text interpretation depends among other things on a pre-processing stage in extracting effectively a correct stem or root. Since there is no available standard stemmer for Arabic, we address here five methods for extracting Arabic roots and the outcomes of the approach with best results will be used later on. Four of these methods are based on a positional-letter-ranking approach where such an approach is investigated along with an adjustment, and two proposed variants. The fifth one is a rule-based approach. An algorithm for correcting irregular words is applied for all methods and a comparison is made between all approaches. The accuracy of these methods was found by comparing extracted roots with a predefined list of roots using an in-house text collection. Results show that the correction algorithm improved the accuracy of the rule-based one by about 14% and the positional letter ranking based algorithms by 7% to 10%. The adjusted positional letter ranking method proved to be the highest in accuracy among all five algorithms but slightly higher than the rule-based one. However, the rule-based algorithm was found to be the approach with the highest accuracy among all ten algorithms when the correction algorithm was included in it.**

Keywords: Arabic Root Extraction; Natural Language Processing; Text Mining; Rule-Based; Positional Letter Ranking; t-test; Variance.

I. INTRODUCTION

Researchers have explored and developed many Text Mining (TM) and Natural Language Processing (NLP) techniques especially to English language but few have been proposed for Arabic text automatic interpretation. This is partially due to the rich morphology [10] of Arabic language. Applying TM techniques requires first a preprocessing stage that would remove punctuation marks, function words and return the remaining words to their stems (for Arabic words to their stems or roots). For English language, researchers perform the stemming step in order to reduce the high dimensionality of documents [12]. Our aim is to compare the performance of different techniques for stemming Arabic words and use the technique with best results. The approach for stemming used here is based on Al-Shalabi, et al work [3] which is a positional letter ranking technique for Arabic root extraction. The choice of this technique was because it is simple, easy to implement and had a reported 90% accuracy. However, in [3] no information were provided of the reasons of choosing the weight and rank values for letters, thus two variant methods of Al-Shalabi, et al work are proposed here along with an adjustment to it. The results of implementing these techniques will be compared with those of a rule-based one thoroughly investigated in [2]. The choice of the rule-based technique was because it reported accuracy higher than 90%. Since the two approaches used here do not handle irregular words, then the ***Correction*** algorithm proposed and implemented in [2] is included in all algorithms and its effectiveness in improving their performance is presented. The importance of handling irregular words comes from the fact that these words presence is about 30% in Arabic texts[1].

The remainder of this paper is organized such that in Section II a brief review, of the various available Arabic root extraction techniques in the literature as well as the Arabic language morphology is introduced. In Section III, the gathered text collection and the function words list are both described. The positional-based-letter ranking techniques and their algorithms along with the rule-based one are presented in Section IV. Section V presents the evaluation criteria and experimental results. Finally, Section VI discusses conclusions and future work.

II. RELATED WORK AND BACKGROUND

A. Related Work

Much [4], [11] work have been performed on Arabic morphological analysis and stemming especially for Information Retrieval (IR) applications. It was concluded that for IR using light stemming provided the highest IR performance followed by that when using root extraction (due to space limitation, the reader is kindly referred to the works in [4] and [11] for excellent reviews). For TM on Arabic, few works, as far as we know, have been conducted to investigate the effect of using stems or roots instead of words on Text Classification (TC) performance such as [14] and [15]. There is a discrepancy among their results. Some reported that light stemming degraded TC performance, whereas others reported that using either stems or roots improved it. It must be noted here that in all mentioned works above for TC, no significance tests were reported.

Many Arabic morphological analysis approaches are rule-based. However, few of such methods handle specific cases of irregular words but not all, to our knowledge, except the works of El-Sadany and Hashish [9] and Beesley [7]. It is note worthy that in El-Sadany and Hashish [9] work, no

1 Percentage values presented here are gathered by 1st author from 40 texts chosen arbitrarily in the collection.

results were provided of the system implemented. Also, in Beesley's work [7], although the Xerox demo[2] is available and efficient, it requires usually a relatively long time to provide the required roots. Thus, there is a need to build an algorithm that provides the correct root for such irregular words. When developing stemmer/morphological analyzer, important issues present themselves [4] such as under-stemming and over-stemming. Other issues that require handling for stemmers are compound words, proper nouns, foreign Arabized words, and irregular forms of words. Here some of the weaknesses presented above are addressed, namely handling irregular words.

B. Background

Arabic language is one of the Semitic languages, [10] that is written from right to left and has 28 letters all consonants: three of these letters are also used as long vowels *A* [3], *w*, *y*. Arabic language has many special cases/properties that affect stemming or any automatic method such as *Hmzp*, short vowels, nunation, and *t$dyd*. These different aspects are highly important in spoken Arabic and in natural language understanding. However, Modern Standard Arabic (MSA) usually does not include short vowels, nunations or assimilation marks in printed texts. Without their presence, the ambiguity of words increases. Verbs are [10] categorized in Arabic as sound or unsound verbs. Unsound verbs are categorized into weak verbs and comprising verbs (either hamzated or geminated).

III. Text Collection and Function Words List

Since no public Arabic text collection is available, an in-house collection of Arabic texts is used to support this work. This collection was gathered, according to eight subject categories, by acquiring arbitrarily Arabic texts from various online Arabic newspapers, academics, magazines and other sources published online in the period 23/7/2008 - 1/2/2009. In each category about 50 texts were chosen randomly with a total of 380 texts (nearly 193,500 words). The Arabic function words list used in this work is formed from 2,549 words [10]. Examples of function words are the separate prepositions, personal pronouns, demonstrative pronouns, relative pronouns, conjunctions, and interjections. Imperfect verbs as *kAn wAxwAthA* were included in the function words list along with similar verbs such as *OSbH* or *mAzAl*. Also, dual and plural forms of the function words are added to their list. Both, constructed text collection and function word list are used in Section V for experiments.

IV. Construction of Root Extractors

In this work, the main purpose is to use/propose variants of a positional-letter ranking approach to extract roots of words in texts as a preprocessing step for TM and to compare the results of such techniques with those of a rule-based one. Both the positional letter ranking one, based on Al-Shalabi, et al [3] work, and the rule-based one, based on Al-Ameed's [1] work concentrate on affix removal of the letters in *sOltmwnyhA*. The original positional-letter ranking technique and the rule-based one presented here do not handle weak, eliminated-long-vowel, hamzated words, names of places, countries, cities, months, broken plurals (except the rule-based one), foreign Arabized words or geminated words (except for the rule-based one where geminating is partially handled and the second variant method). So, the ***Correction*** algorithm [2] is used here in all techniques in order to improve their performance and investigate its effectiveness. The approach with the best performance results will be used in TM procedures later on. The performance of the techniques before/after adding the ***Correction*** algorithm to them will be presented in Section V.

A. Non-Linguistic Approach

This approach uses the positional-letter-ranking work proposed by Al-Shalabi, et al [3][4] and a slight adjustment to it along with two proposed variants to it. The above techniques test at the beginning if the number of letters in the word is less than or equal to 3 and if so take the word (except for the fourth technique) without any further processing. The fourth technique tests if a two-letter word is geminated by comparing it to a two-letter geminated words list. If it is in the list, the fourth technique presents the two-letter word as a triliteral root by doubling its second letter. Also, the third and fourth techniques extract specific cases of quadriliteral roots along with triliteral ones whereas the first two methods extract only triliteral roots. Section V presents the outcome of implementing these techniques.

Al-Shalabi Algorithm. This algorithm employs a letter weight, an order index and assigns a rank to a letter according to its order in the word. ***Al-Shalabi*** algorithm extracts the root for the word through the following simple steps: 1- for each letter in the word (from right to left) apply weight and rank values according to Tables I and II while assigning order values, 2- calculate the product of the rank and weight for each letter, 3- keep only the letters with the first three smallest product values without changing the order of these letters in the word. In order to illustrate the performance of this algorithm, two examples of words are shown in Table III where the least three product values are bolded. As shown in Table I, the rank of a word is calculated differently when its number of letters is odd from that when it is even. The weights of letters are given values for letters categorized into groups (e.g. allocating the group of letters '*p*, *A*' a weight of 5) as shown in Table II. Al-Shalabi, et al work did not explain or clarify why or on what basis did it use such ranking or weighting only that such groups and their values were chosen after extensive experimentation.

2 Xerox demo can be found at http://www.xrce.xerox.com/Research-Development/Historical-projects/Linguistic-Demos/Arabic-Morphological-Analysis-and-Generation

3 Transliterations used here are that of Buckwalter' found at: http://www.qamus.org/transliteration.htm

4 Many thank goes to H. Al-Serhan for providing a copy of Al-Shalabi, et al (2003) paper.

TABLE I. LETTER RANKING IN AL-SHALABI ALGORITHM *(DERIVED FROM (AL-SHALABI, ET AL 2003) WORK)*

Letter position from right	Rank (if word length is even)	Rank (if word length is odd)
1	N	N
2	N – 1	N – 1
3	N – 2	N – 2
:	:	:
$\lceil N/2 \rceil$	N/2 + 1	$\lceil N/2 \rceil$
$\lceil N/2 \rceil + 1$	N/2 + 1 – 0.5	$\lceil N/2 \rceil + 1 - 1.5$
$\lceil N/2 \rceil + 2$	N/2 + 2 -0.5	$\lceil N/2 \rceil + 2 - 1.5$
$\lceil N/2 \rceil + 3$	N/2 + 3 -0.5	$\lceil N/2 \rceil + 3 - 1.5$
:	:	:
N	N – 0.5	N – 1.5

Where N: number of letters in a word

TABLE II. TABLE 2: WEIGHTS OF LETTER GROUPS IN AL-SHALABI ALGORITHM *(DERIVED FROM (AL-SHALABI, ET AL 2003) WORK)*

Letters	*A, p*	*y, }*	*t, w, Y*	*O, I, m, n*	*l, s, h*	Rest
Weight	5	3.5	3	2	1	Zero

TABLE III. EXAMPLES OF EXTRACTED ROOTS USING [3]

a) word *IstxdAmh*, correct root *xdm*

letters	*h*	*m*	*A*	*d*	*x*	*t*	*s*	*I*
Order	8	7	6	5	4	3	2	1
Weight	1	2	5	0	0	3	1	2
Rank	7.5	6.5	5.5	4.5	5	6	7	8
Product	7.5	13	27.5	**0**	**0**	18	**7**	16
Root	*sxd* (X)							

b) word *AltElymAt*, correct root *Elm*

Letters	*t*	*A*	*M*	*y*	*l*	*E*	*t*	*l*	*A*
Order	9	8	7	6	5	4	3	2	1
Weight	3	5	2	3.5	1	0	3	1	5
Rank	7.5	6.5	5.5	4.5	5	6	7	8	9
Product	22.5	32.5	11	15.75	**5**	**0**	21	**8**	45
Root	*lEl* (X)								

Adjusted Al-Shalabi Algorithm. It was noticed in [3] that there was a discrepancy in some of its examples. The two examples that caused such discrepancy were the ones when the letter *l* was at first or second positions of a word where the authors have given it a weight of 5. However, it was given a weight of 1 when it was in other positions (as already specified in their paper). This information was not explained or mentioned throughout the paper except only in the two examples. So, here this is considered a possible adjustment algorithm (named ***Adjusted Al-Shalabi***) while maintaining the rest of the procedure mentioned in [3]. Thus, the same ranking and ordering of letters in a word were maintained, and a different weight of 5 to only the letter *l* was given if it was in the first or second positions in the word. Following this adjustment of the weight of the letter *l* when applied on the same two examples in Table III, the expected extracted roots would be *sxd* and *Elm* respectively.

As can be seen from the examples in the previous two techniques, it is expected that these algorithms will not extract roots with high accuracy. However, since this approach is very simple and easy to implement, then proposing a different weighting scheme, based on considering the characteristics of occurrence of Arabic language letters, for the groups of letters might produce higher accuracy results. Statistics showing the percentages of such Arabic letters were found[5]. After close examination of these percentages and including the effect of the number of letters before and after them as shown in Table IV, it was not possible to quantitatively reach a weight for these letters or classify them into separate distinct groups. However, it was possible to do so qualitatively: 1- At a first analysis, it was proposed that these letters be grouped into 5 groups (as ***Al-Shalabi*** algorithm or its adjustment) where here such groups are assigned classes: *high, high or moderate, moderate, moderate or low,* and finally *low*. 2- However, at a second analysis; it was proposed that these letters be grouped into 4 groups where here such groups are assigned classes: *high, high or moderate, moderate, and finally moderate or low*. 3- At a third analysis, it was proposed that these letters be grouped into 3 groups where here such groups are assigned classes: *high, moderate, and finally moderate or low*. The reason why no conclusive number of groups was reached is the nature of some of these letters and their similar percentages in appearing as extra and original letters in words. Also, it was not possible to quantitatively find all the weights proposed by ***Al-Shalabi*** algorithm from these statistics. However, since the initial number of groups found here are 5, weighting letters was thus given by assigning the groups weights from 5 to 1 according to classes assigned: 5 for *high*, 3.5 for *high or moderate*, 3 for *moderate*, and so on. In order to investigate these different choices of the number of groups, the first *Variant* method explained next will adapt grouping these letters into 4 groups, while the second *Variant* method will group such letters into 3 groups.

TABLE IV. PERCENTAGES OF LETTERS APPEARANCE IN TEXTS

Letter	Rate (%)	no. of letters after & %	no. of letters before & %	letter after with highest %	letter before with highest %	Qualitative weight for rate values only
Space	**not given**	**27 84 OL**	**32 100 OL**	**43.02 A**	21.9p	**not a character**
A	19.65	3 1 96 OL	30 94 OL	40.81	space 42.16	**high**
p	4.22	1 3 OL	30 84 OL	space 100	y 38.41	**moderate or low**
h	1.79	1 4 44 OL	22 69 OL	A 41.03	15.8 1 – t	**Low**
}	0.50	1 0 31 OL	6 19 OL	y 41.54	52.31 A	**Low**
y	6.66	2 8 88 OL	31 97 OL	space 25.49	16.07 f	**high or moderate**
l	12.99	3 0 94 OL	29 91 OL	A 21.24	A 61.71	**high**
t	5.64	3 1 97 OL	24 75 OL	space 22.76	A 22.49	**moderate**
w	5.70	3 0 94 OL	27 84 OL	16.09 **A**	space 41.02	**moderate**
Y	0.91	1 3 OL	9 28 OL	space 100	1 71.43	**Low**
m	8.52	3 0 94 OL	25 78 OL	space 19.73	1 22.33	**high or moderate**
n	3.86 %	25 78 OL	21 66 OL	space 42.57%	A 28.12	**moderate or low**

5 From Khaled AlShamaa web site, URL: http://www.alshamaa.com/php/arabic/index.html, [last accessed: 4/6/2010]

s	2.48 %	20 63 OL	17 53 OL	%20 1	28.92 1	**moderate or low**

Where . ***OL*** *STANDS FOR* ***O****F* ***L****ETTER*

First Proposed Variant of Al-Shalabi Algorithm. Here, it is proposed to use the same ranks of letters as that of ***Al-Shalabi*** algorithm but to assign a different set of weights to letters as shown in Table V in order to provide a triliteral root according to their order. The five groups of letters that were proposed in [3] have been reduced to four with the shown weights. The letter *l* was moved to third group with weight 3. The letter *s* was moved to the fourth group to give it a higher value especially when at beginning of a word (most likely it will be an extra letter but an original letter elsewhere). Finally, the letter ***h*** was moved to the first group with weight 5 since it is expected that when ***h*** is at the end of the word, it is likely to be a suffix since it might be wrongly written as ***h*** where as it is meant to be *p*. This algorithm is called ***Variant1***. Also, this algorithm proposes to extract specific cases of quadriliteral-root-based words. Since ***Variant1*** algorithm is a combination of the original positional letter ranking method and rules to handle quadriliteral roots, then it is a hybrid method. The original two examples in Table III would generate the roots *sxd* and *Elm* respectively, when using this algorithm.

TABLE V. WEIGHTS OF LETTER GROUPS FOR *VARIANT1* ALGORITHM

Letters	*A, p, h*	*y, }*	*l, t, w, Y*	*O, I, m, n, s*	**Rest**
Weight	5	3.5	3	2	Zero

Variant1 Algorithm

```
Inputs: Set of preprocessed documents D = {d1, d2, ....,dn}, Predefined root lists, Predefined letter groups weight lists
Outputs: List of triliteral and some quadriliteral roots for each new document new_di_1
START
For each document di do {
LastWord = Count_No_Words(di)
For j = 1 to LastWord in di do {
LastLetter = Count_No_Letters(wj,c)
If(LastLetter <= 3){Final_Wordj = wj, go to *}
Provide the order, weight values for each letter in word wj
Perform calculating the product of order and weight values for each letter in word wj
Count = Count_No_Zero_Product_Letters(wj)
If((Count > 3) and (LastLetter >= 4)){ Final_Wordj=Extract_4letter_with_least_product (wj), go to *} Else
{ Final_Wordj=Extract_3letter_with_least_product(wj )}
* Write Final_Wordj to output document new_di_1
Calculate Accuracy_of_document_ new_di_1 }
END
```

In brief, this algorithm varies from previous ones by providing different groups of letters with different weight values and extracting four-letter roots.

Second Proposed Variant Algorithm. The second variant technique of ***Al-Shalabi*** algorithm (that is named here ***Variant2***) uses the same ranks as described in [3]. This second technique performs the following steps: 1- it excludes the letter combination *Al* (i.e. the definitive article) from the word if the word starts with it, 2- it replaces the letters *O, I, |* with *A* and replaces letters *}, y* with *Y* and replaces letter *p* with *h* (i.e. a normalization step), 3- it presents specific two-letter geminated words as triliteral by comparing them with a predefined list of two-letter geminated words and if the two-letter word is in the list, the algorithm duplicates the second letter, 4- it uses a different weighting scheme as shown in Table VI other than the previous three techniques, 5- it provides a quadriliteral root by counting the number of zero product values for letters in a word (other than the letter *b*) and by counting the number of repetitions a letter occurs in a word (other than the letters *b* or *w* or *A*). As can be noticed from the algorithm, more rules were put for choosing a quadriliteral root. This is due to the fact that in some four-letter words using ***Variant1*** algorithm, these words will be considered as a correct root where they are not. As can be seen from Table VI, the five groups of letters that were proposed in [3] have been reduced to only three with the shown weights. Here, the second group (in Table II) is cancelled since its letters are replaced by *Y*. Also, the letters *l, m, s* and *n* are moved to the third group with weight 2, and the letters *t, w* and *Y* were moved to the second group with weight 3. The original two examples, from Table III, using this variant would extract roots, *xdm* and *Elm* respectively.

TABLE VI. WEIGHTS OF LETTER GROUPS FOR *VARIANT2* ALGORITHM

Letters	*A, h*	*t, w, Y*	*l, m, n, s*	Rest
Weight	5	3	2	Zero

Variant2 Algorithm

```
Inputs: Set of preprocessed documents D = {d1, d2, ....,dn}, Predefined root lists, Predefined two-letter geminated words list, 3 Predefined Replace lists, Predefined letter groups weight lists
Outputs: List of triliteral and some quadriliteral roots for each new document new_di_1
START
For each document di do {
LastWord = Count_No_Words(di)
For j = 1 to LastWord in di do {
LastLetter = Count_No_Letters(wj,c)
If(LastLetter < 3){{Final_Wordj = wj, go to *}}
Remove_AL(wj)
Replace_letters(wj) % a normalization step
Provide the order, weight for each letter in wj.
Perform calculating the product of weight and order values for each letter in word wj.
Count = Count_No_Zero_Product_Letters_Not_b(wj)
Repeat = Count_No_Repetitions_Not_b_w_A(wj)
If(((Count > 3) or (Repeat > 2)) and (LastLetter >= 4))
{Final_Wordj=Extract_4letter_with_least_product(wj), go to *} Else
{ Final_Wordj=Extract_3letter_with_least_product(wj )}
* LastLetter = Count_No_Letters(Final_Wordj,c)
If (LastLetter == 2) {{cc = Compare (Final_Wordj, 2_letter_list)}
If(cc==0){Final_Wordj=Correct_Word(Final_Wordj)}}
Write Final_Wordj to output document new_di_1
Calculate Accuracy_of_document_ new_di_1}
END
```

In brief, this algorithm varies from the previous ones by that it: 1- provides different weight values for different groups of letters, 2- removes *Al* from words if these words start with it, 3- perform a normalization step, 4- handles two-letter geminated roots, and 5- extracts four-letter roots.

B. The Rule-Based Approach

The rule-based stemmer is implemented starting from the work of Al-Ameed [1]. It is composed of two parts: a rule-based light stemmer, and a pattern-based infix remover. The rule-based light stemmer removes prefixes and suffixes from the word according to specific rules. The pattern-based infix remover removes infixes from the word according to specific patterns. This approach is named here ***Rule-Based*** algorithm. The basic steps of this algorithm is simple: 1- it stems the word if its number of letters is greater than 3, 2- it outputs this stem to a new document, 3- it performs infix removal on this stem, 4- it outputs the resulting root to another new document and calculates the accuracy of algorithm after all words are processed from the input document. When the ***Correction*** algorithm is included, step 4 is modified such that the algorithm corrects irregular triliteral roots (if extracted root is not found in the root list) and then performs the remainder. This root extractor was explained in details in [2] and due to space limitation, the reader is kindly referred to it for more details. All techniques are evaluated using accuracy which is found by each algorithm by comparing each extracted root with a predefined list of 5,405 roots that contains lists of only triliteral and quadriliteral roots (4,655 triliteral roots and 750 quadriliteral roots collected from ([5], [6], and [8])), then counting the roots that match the ones in the predefined list, and finally calculating the percentage of correct roots in each text of the collection.

V. EXPERIMENTAL RESULTS AND ANAYLSIS

A. Accuracy Results

The accuracy of algorithms is shown in Fig. 1 along with their ***Correction*** counterparts[6]. In Fig. 1, the ***Adjusted Al-Shalabi*** and the ***Rule-Based*** algorithms provided the highest accuracy results (with or without the correction algorithm). The effect of adding the ***Correction*** algorithm to all techniques was to increase their accuracy by about 7% to 14%. Although ***Variant1*** algorithm is higher in accuracy than ***Variant2*** algorithm, nevertheless, when their ***Correction*** algorithm is added, the opposite occurs. This result indicates that ***Variant2*** algorithm is more sensitive to irregular words. Also, the ***Rule-Based*** algorithm is rather less in accuracy than the ***Adjusted Al-Shalabi*** algorithm by about 2.2%. However, the ***Rule-Based Correction*** algorithm's accuracy is rather higher than that the ***Adjusted Al-Shalabi Correction*** algorithm's accuracy by about 2%. As can be seen from the results [13, pp. 208 – 210], the differences among such algorithms are rather small which requires calculating variance using (1) for all algorithms:

$$Var = \sum_{i=1}^{n} (x_i - \bar{x})^2 \qquad (1)$$

Where n: number of texts, x_i: accuracy of i^{th} text, $\bar{x}$: average accuracy of n texts.

The results of variance for all algorithms in all categories are shown in Fig. 2. The variance values for the ***Adjusted Al-Shalabi*** and ***Rule-Based*** algorithms along with their ***Correction*** are shown in Fig. 3 where these values are very near and can not clarify which of the two algorithms (or their ***Correction*** ones) is better. Thus, [13, pp. 208 – 210] using (2) t-test is found by hypothesizing that ***Adjusted Al-Shalabi*** algorithm is better than ***Rule-Based*** algorithm (as the null hypothesis). The t-value was found to be 5.56 and for α = 0.01, the critical value of t is 2.576 (using a one-tailed test with ∞ degrees of freedom [13, pp. 609]). Since t = 5.56 > 2.576 then the hypothesis is accepted here.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{2s^2}{n}}}, \text{ where } .... \; s^2 = \frac{Var_1 + Var_2}{n_1 + n_2 - 2} \qquad (2)$$

Where** $\bar{x}_1$: accuracy of Adjusted Al-Shalabi algorithm, $\bar{x}_2$: accuracy of Rule-Based algorithm, s^2: pooled variance of both algorithms,* and Var_1: variance of ***Adjusted Al-Shalabi algorithm, Var_2: variance of ***Rule-Based*** algorithm, $n_1 = n_2$: number of texts for both algorithms.*

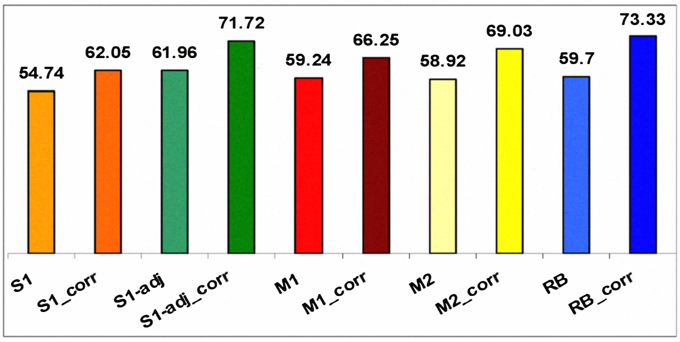

Figure 1. Comparison between accuracy results of all ten algorithms

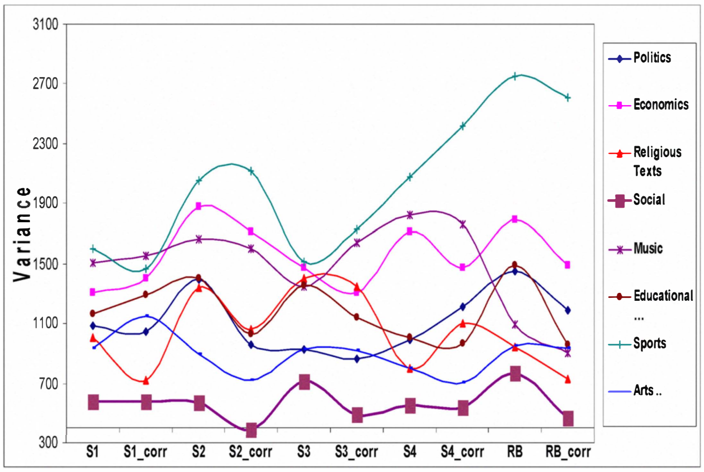

Figure 2. Variance values for all algorithms among all categories *(points were connected here by smooth curves for illustration purposes only)*

6 S1: Al-Shalabi algorithm, S1_corr: Al-Shalabi with Correction algorithm, S2: Adjusted Al-Shalabi algorithm, S2_corr: Adjusted Al-Shalabi with Correction algorithm, S3: Variant1 algorithm, S3_corr: Variant1 with Correction algorithm, S4: Variant2 algorithm, S4_corr: Variant2 with Correction algorithm, RB: Rule-Based algorithm, RB_corr: Rule-Based with Correction algorithm.

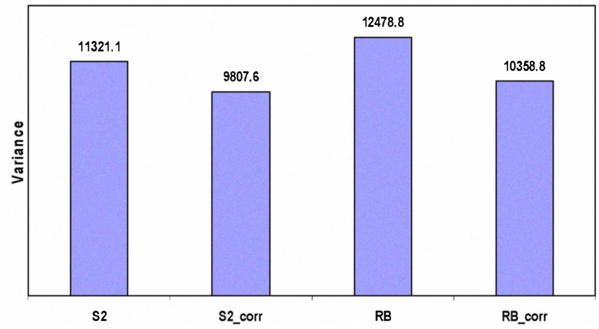

Figure 3. Comparison between variance values for *Rule-Based* and *Adjusted Al-Shalabi* algorithms along with their *Correction* ones

The t-test is also performed for the two algorithms with the ***Correction*** algorithm where the null hypothesis here is that ***Rule-Based Correction*** algorithm is better than the ***Adjusted Al-Shalabi Correction*** algorithm. Using the same equations as above and for $\alpha = 0.01$, the hypothesis is accepted. Thus, one concludes that the approach with highest accuracy among all algorithms is the rule-based approach (with the ***Correction*** algorithm). Also, although not shown here, the ***Correction*** algorithm, in general, lowered variance and improved performance of all algorithms and categories. Also, an in-coder[7] analyzed their performance and found that the ***Correction*** algorithm relatively improved the rule-based one by about 10% whereas it relatively improved the positional letter ranking techniques by about 5.2% – 5.5%. Results of in-coder analysis are shown in Fig. 4 and Fig. 5 and are not near those reported in Fig. 1. This is due to some limitations as: 1- in specific cases the correction algorithm does not check the extracted root since it is not reached[8]. 2- In other cases the extracted root is not found in the algorithm to be corrected since its case is not handled. 3- In other cases the extracted root is not found in the root list since the root list provided here does not include all roots (estimated 10,000 roots [4]). 4- In other cases, although relatively few, a surface word might have more than one option for correction and the algorithm chooses (according to its structure) only one of them (that might be wrong).

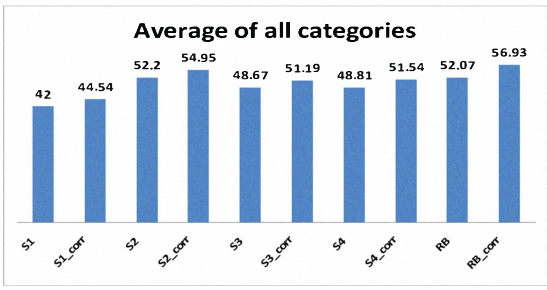

Figure 4. Native Arabic speaker analysis of algorithm' accuracy

7 The 1st author of this paper. This was done as a preliminary step using 40 texts (5 from each class chosen arbitrarily).

8 This is due to the fact that the extracted root is found in the predefined root list (so is considered correct even though it is actually the wrong root).

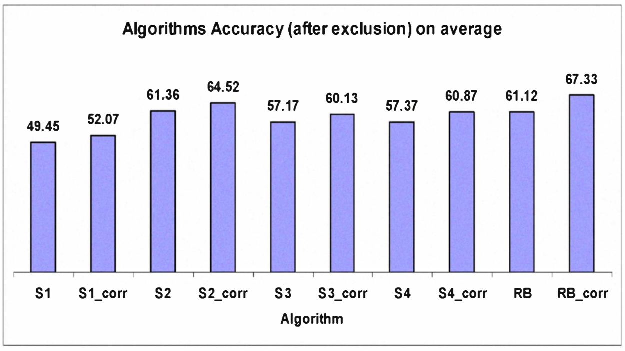

Figure 5. Native Arabic speaker analysis of algorithm' accuracy after excluding number of names, transliterations, stop words and compounds from total number of words in texts

VI. CONCLUSIONS AND FUTURE WORK

A positional letter ranking approach for root extraction was investigated. Two variants along with an adjustment to it were also proposed and implemented here. The results of implementing such techniques were compared with those of a rule-based one. It was found that the ***Correction*** algorithm do indeed improve the performance of the two approaches. The ***Adjusted Al-Shalabi*** method proved to be the highest in accuracy among all five original algorithms. However, the ***Rule-Based*** algorithm became the approach with the highest accuracy among all ten algorithms when the ***Correction*** algorithm was included in it (improvement by about 14%). The experiments show a promising future for the proposed ***Correction*** algorithm to be implemented for other stemmers. Yet, it has some limitations and the 14% improvement can be increased by adding further rules and restrictions. Also, the ***Adjusted Al-Shalabi*** method can be further improved by: a) handling two-letter geminated words, performing normalization to handle weak words, and extract some quadriliteral-root based words (as was proposed in the second variant), b) handling the special effect of *b* as an extra letter (when at beginning of a word) and c) proposing a range of weight values instead of specific ones. This is so since it is clear from the experimental results that the two proposed grouping of letters and their respective weights did not provide in general higher accuracy values. However, it was observed that each proposed method gave the correct root for some words but failed for others, while the ***Adjusted Al-Shalabi*** method provided the correct root for many others. This suggests that using a range of weight values to such letters might provide higher accuracies (instead of specific values). Also, acquiring larger text collection would emphasize the results of the performance of such techniques.

ACKNOWLEDGMENT

The first author would like to thank Petra University, Amman, Jordan for partially financing her PhD study.

REFERENCES

[1] H.K Al-Ameed. A proposed new model using a light stemmer for increasing the success of search in Arabic terms. PhD Thesis, Bradford, UK: University of Bradford. 2006.

[2] M. Y. Al-Nashashibi, D. Neagu, A. Yaghi. "An improved root extraction technique for Arabic words". In 2nd Int. Conference on Computer Technology and Development ICCTD 2010, 2-4 November 2010, in press.

[3] R. Al-Shalabi, G. Kannan, and H. Al-Serhan. "New Approach For extracting Arabic roots". In Proc of 2003 International Arab conference on Information Technology (ACIT'2003), Alexandria, 2003, pp. 42-59.

[4] I.A. Al-Sughaiyer and I.A Al-Kharashi. "Arabic morphology analysis techniques: A comprehensive survey". J. of the American Society for Information Science & Technology JASIST. Feb. 2004, 55(3), pp. 189 – 213.

[5] Imam Mohammed Ibn Abi Baker Ar-Rhazi. Mukhtar us-Sihah. Beirut: Librairie du Liban Publishers. 1986. (in Arabic).

[6] H. Bayyomee, Kh. Kolfat, and A. Al-Shafe'e. Lexicon for Arabic verbs morphology. Cairo: Dar Ilias Modern Publishing Comp. 1989. (in Arabic).

[7] K.R. Beesley. "Finite-State morphological analysis and generation of Arabic at Xerox Research: status and plans in 2001". In: ARABIC NLP Workshop: Status and Prospects ACL-EACL2001, Toulouse, France 6 July 2001, pp. 1 – 8.

[8] A. El-Dahdah. A Dictionary of Arabic grammar in charts and tables. Beirut: Librairie du Liban Publishers. 2008. Revised by: Dr. GM Abdul-Massih (in Arabic).

[9] T. A. El-Sadany and M. A. Hashish. "An Arabic morphological system". IBM Systems Journal, 28(4), 1989, pp. 600-612.

[10] J.A. Haywood and H.M. Nahmad. A new Arabic grammar of written language. London: Lund Humphries Publishers. 1998.

[11] Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell. "Light stemming for Arabic Information retrieval". In Abdelhadi Soudi, Antal van den Bosch and Gunter Neumann, Eds., Arabic computational morphology knowledge-based and empirical methods, text, speech and language technology series, Vol. 38, Part IV, pp. 221 – 243, The Netherlands: Springer , 2007.

[12] F. Sebastiani, and Nazionale Delle Ricerche Consiglio. "Machine learning in Automated Text Categorization". ACM Computing Survey, 34(1), 2002, pp. 1 – 47.

[13] C.D. Manning and H. Schütze. Foundations of statistical natural language processing. Massachusetts, USA: MIT press. 1999.

[14] A. M. Mesleh. "Chi Square feature extraction based SVMs Arabic language text categorization system". J. Computer Science, 3(6), Science Publications, pp. 430-435, 2007.

[15] S. Raheel, J. Dichy, and M. Hassoun. "The automatic categorization of Arabic documents by boosting Decision Trees". Proceedings of 5th Int. Conference on Signal-Image Technology and Internet-based Systems SITIS 2009, 29 Nov – 4 Dec 2009, Marrakech, Morocco. IEEE Xplore.